**BST Iterator**

#include <bits/stdc++.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Following is the TreeNode class structure

template <typename T>

class TreeNode {

public:

T data;

TreeNode<T> \*left;

TreeNode<T> \*right;

TreeNode(T data) {

this->data = data;

left = NULL;

right = NULL;

}

};

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class BSTiterator

{

//private:

stack<TreeNode<int>\*>st;

public:

BSTiterator(TreeNode<int> \*root)

{

// write your code here

pushAll(root);

}

int next()

{

// write your code here

TreeNode<int>\* node = st.top();

st.pop();

pushAll(node->right);

return node->data;

}

bool hasNext()

{

// write your code here

return !st.empty();

}

void pushAll(TreeNode<int>\* root){

for(;root!= NULL;st.push(root), root= root->left);

}

};

/\*

Your BSTIterator object will be instantiated and called as such:

BSTIterator iterator(root);

while(iterator.hasNext())

{

print(iterator.next());

}

\*/